

合肥工业大学

操作系统实验报告

实验题目 实验 13 扫描 FAT12 文件
系统管理的软盘

学生姓名 袁焕发

学 号 2019217769

专业班级 物联网工程 19-2 班

指导教师 田卫东

完成日期 2021 年 11 月 24 日

1. 实验目的和任务要求

通过查看 FAT12 文件系统的扫描数据，并调试扫描的过程，理解 FAT12 文件系统管理软盘的方式。
通过改进 FAT12 文件系统的扫描功能，加深对 FAT12 文件系统的理解。

2. 实验原理

EOS 使用的 FAT12 文件系统中，一个簇只包含一个扇区，所以扇区的分布可以如图 8-2 所示。

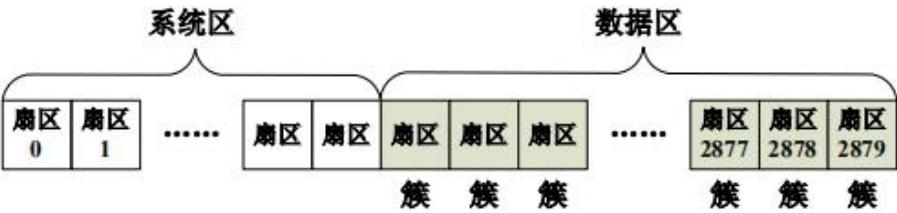


图 8-2：1.44M 软盘上 FAT12 文件系统的扇区分布

FAT12 文件系统的系统区从 0 扇区开始，到 32 扇区结束。这 33 个扇区又分成了三部分，分别是引导扇区、文件分配表（FAT）和根目录。其中引导扇区占用 0 扇区，FAT 表占用 1 到 18 扇区，根目录占用 19 到 32 扇区。余下的 33 到 2879 扇区是数据区。有了这些详细的分布信息，图 8-2 就可以演变为图 8-3 了。

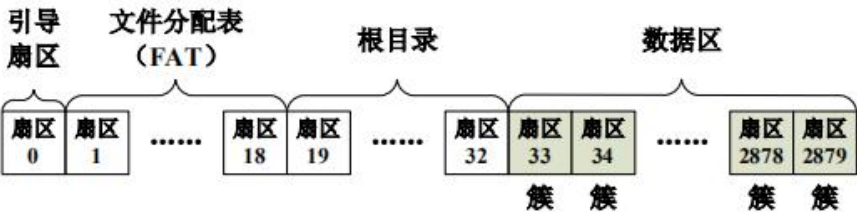


图 8-3：1.44M 软盘上 FAT12 文件系统的更详细的扇区分布

3. 实验内容

3.1. 阅读控制台命令“sd”相关的源代码，并查看其执行的结果

新建一个 EOS Kernel 项目，按 F7 生成在本实验 3.1 中创建的 EOS Kernel 项目，启动调试。在 EOS 控制台中输入命令“sd”后按回车。

```
Bochs for Windows - Display
A: B: CD: USER Copy Paste Snapshot T1 Reset Run/Stop Power
CONSOLE-1 (Press Ctrl+F1-F4 to switch console window...)
>sd
***** BIOS Parameter Block (BPB) *****
Bytes Per Sector : 512
Sectors Per Cluster : 1
Reserved Sectors : 1
Fats : 2
Root Entries : 224
Sectors : 2880
Media : 0xF0
Sectors Per Fat : 9
Sectors Per Track : 18
Heads : 2
Hidden Sectors : 0
Large Sectors : 0
***** BIOS Parameter Block (BPB) *****

First Sector of Root Directory : 19
Size of Root Directory : 7168
First Sector of Data Area : 33
Number Of Clusters : 2847

Free Cluster Count: 2275 (1164800 Byte)
Used Cluster Count: 572 (292864 Byte)
>
```

3.2. 根据 BPB 中的信息计算出其他信息

修改“sd”命令函数 ConsoleCmdScanDisk 的源代码，在输出 BPB 中保存的信息后，不再通过 pVcb->FirstRootDirSector 等变量的值进行打印输出，而是通过 BPB 中保存的信息重新计算出下列信息，并打印输出：

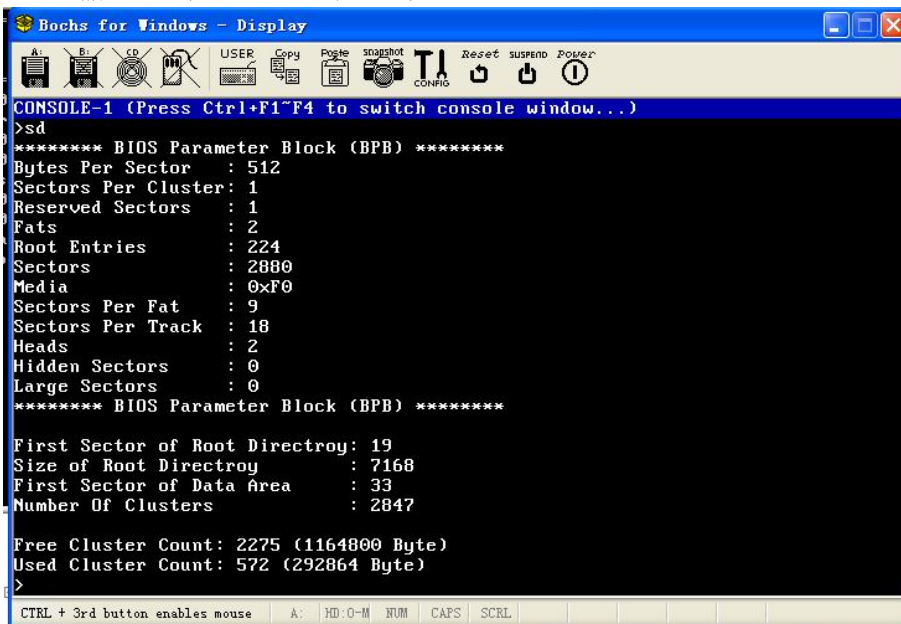
- (1) 计算并打印输出根目录的起始扇区号，即 pVcb->FirstRootDirSector 的值。
- (2) 计算并打印输出根目录的大小，即 pVcb->RootDirSize 的值。
- (3) 计算并打印输出数据区的起始扇区号，即 pVcb->FirstDataSector 的值。
- (4) 计算并打印输出数据区中簇的数量，即 pVcb->NumberOfClusters 的值。

```
// 将卷控制块中缓存的 BIOS Parameter Block (BPB)，以及卷控制块中的其它重要信息输出
//
fprintf(StdHandle, "***** BIOS Parameter Block (BPB) *****\n");
fprintf(StdHandle, "Bytes Per Sector : %d\n", pVcb->Bpb.BytesPerSector);
fprintf(StdHandle, "Sectors Per Cluster: %d\n", pVcb->Bpb.SectorsPerCluster);
fprintf(StdHandle, "Reserved Sectors : %d\n", pVcb->Bpb.ReservedSectors);
fprintf(StdHandle, "Fats : %d\n", pVcb->Bpb.Fats);
fprintf(StdHandle, "Root Entries : %d\n", pVcb->Bpb.RootEntries);
fprintf(StdHandle, "Sectors : %d\n", pVcb->Bpb.Sectors);
fprintf(StdHandle, "Media : 0x%X\n", pVcb->Bpb.Media);
fprintf(StdHandle, "Sectors Per Fat : %d\n", pVcb->Bpb.SectorsPerFat);
fprintf(StdHandle, "Sectors Per Track : %d\n", pVcb->Bpb.SectorsPerTrack);
fprintf(StdHandle, "Heads : %d\n", pVcb->Bpb.Heads);
fprintf(StdHandle, "Hidden Sectors : %d\n", pVcb->Bpb.HiddenSectors);
fprintf(StdHandle, "Large Sectors : %d\n", pVcb->Bpb.LargeSectors);
fprintf(StdHandle, "***** BIOS Parameter Block (BPB) *****\n\n");

fprintf(StdHandle, "First Sector of Root Directroy: %d\n", pVcb->Bpb.ReservedSectors+pVcb->Bpb.Fats*pVcb->Bpb.SectorsPerFat);
fprintf(StdHandle, "Size of Root Directroy : %d\n", pVcb->Bpb.RootEntries*32);
fprintf(StdHandle, "First Sector of Data Area : %d\n", pVcb->Bpb.ReservedSectors+pVcb->Bpb.Fats*pVcb->Bpb.SectorsPerFat);
fprintf(StdHandle, "Number Of Clusters : %d\n", pVcb->Bpb.Sectors-((pVcb->Bpb.ReservedSectors+pVcb->Bpb.Fats*pVcb->Bpb.SectorsPerFat)));

// 扫描 FAT 表，统计空闲簇的数量，并计算软盘空间的使用情况
//
FreeClusterCount = 0;
for (i = 2; i < pVcb->NumberOfClusters + 2; i++) {
    if (0 == FatGetFatEntryValue(pVcb, i))
        FreeClusterCount++;
}
UsedClusterCount = pVcb->NumberOfClusters - FreeClusterCount;
fprintf(StdHandle, "Free Cluster Count: %d (%d Byte)\n", FreeClusterCount, FreeClusterCount*pVcb->Bpb.SectorsPerCluster*pVcb->Bpb.BytesPerSector);
fprintf(StdHandle, "Used Cluster Count: %d (%d Byte)\n", UsedClusterCount, UsedClusterCount*pVcb->Bpb.SectorsPerCluster*pVcb->Bpb.BytesPerSector);
```

源代码修改完毕后，按 F7 生成项目，启动调试。待 EOS 启动完毕，在 EOS 控制台中输入命令“sd”后按回车。



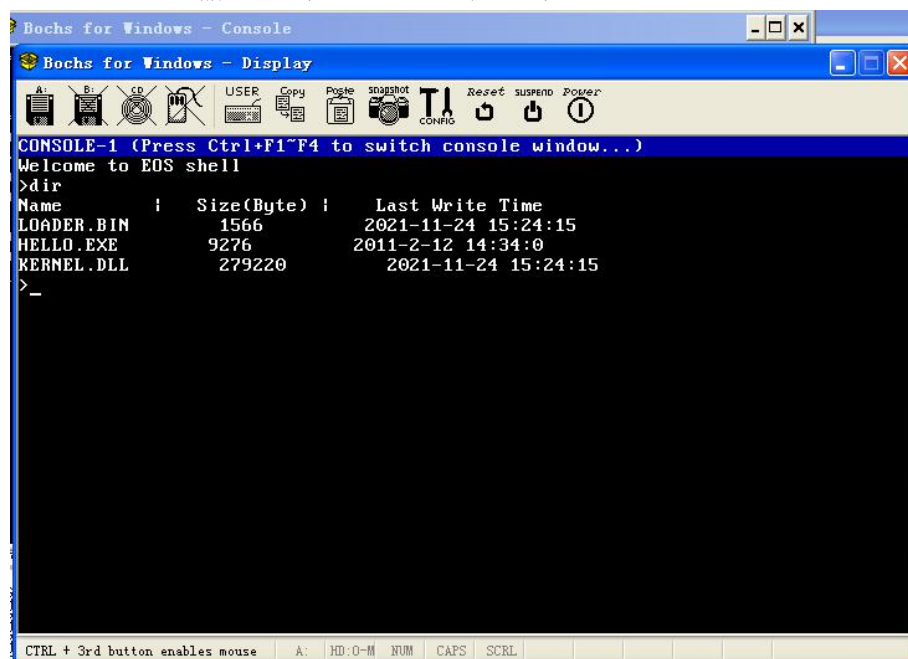
```
Bochs for Windows - Display
USER Copy Paste Snapshot CONFIG Reset SUSPEND POWER
CONSOLE-1 (Press Ctrl+F1~F4 to switch console window...)
>sd
***** BIOS Parameter Block (BPB) *****
Bytes Per Sector : 512
Sectors Per Cluster: 1
Reserved Sectors : 1
Fats : 2
Root Entries : 224
Sectors : 2880
Media : 0xF0
Sectors Per Fat : 9
Sectors Per Track : 18
Heads : 2
Hidden Sectors : 0
Large Sectors : 0
***** BIOS Parameter Block (BPB) *****

First Sector of Root Directroy: 19
Size of Root Directroy : 7168
First Sector of Data Area : 33
Number Of Clusters : 2847

Free Cluster Count: 2275 (1164800 Byte)
Used Cluster Count: 572 (292864 Byte)
>
```

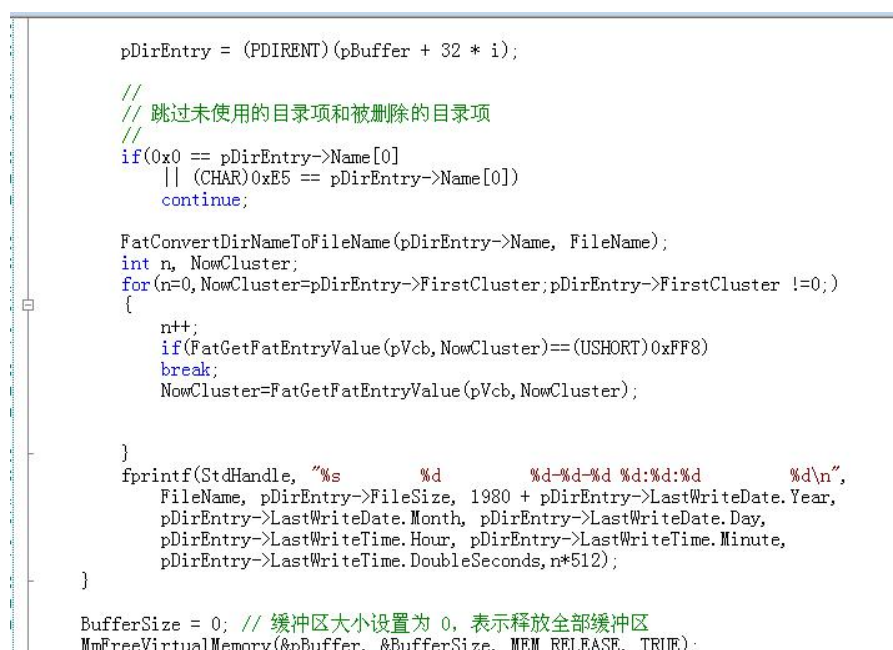
3.3. 阅读控制台命令“dir”相关的源代码，并查看其执行的结果

按 F7 生成创建的 EOS Kernel 项目。按 F5 启动调试。待 EOS 启动完毕，在 EOS 控制台中输入命令“dir”后按回车。



3.4. 输出每个文件所占用的磁盘空间的大小

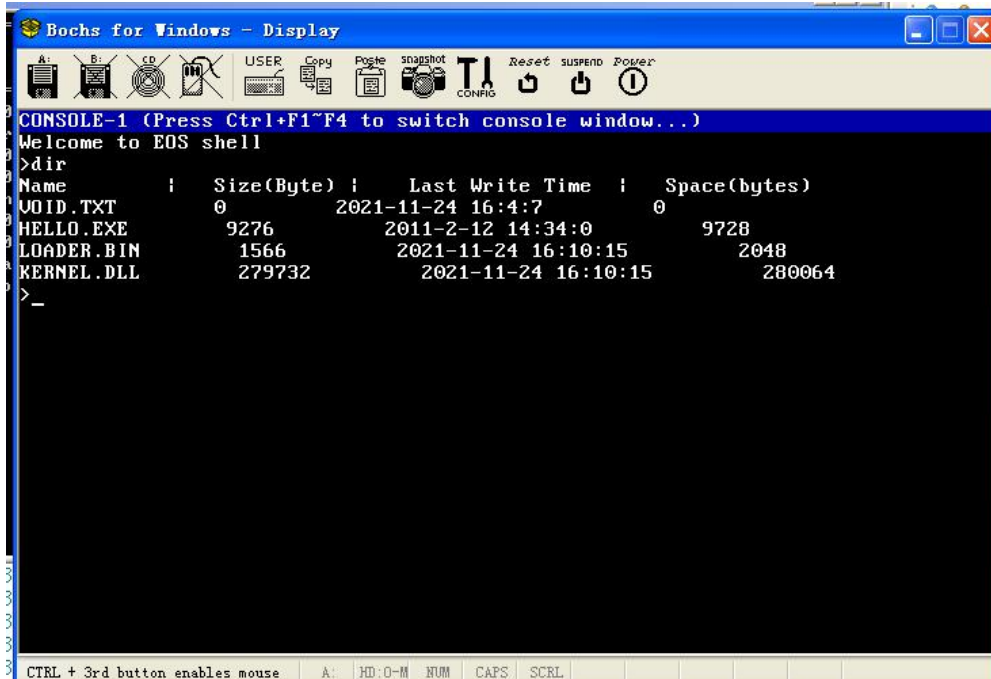
修改“dir”命令函数 ConsoleCmdDir 的源代码，要求在输出每个文件的名称、大小、最后改写时间后，再输出每个文件所占用的磁盘空间（以字节为单位）。



在“项目管理器”窗口中双击 Floppy.img 文件，使用 FloppyImageEditor 工具打开此软盘镜像。将“学生包”文件夹中的 void.txt 文件（大小为 0）添加到软盘镜像的根目录中（将 void.txt 文件拖动到 FloppyImageEditor 窗口中释放即可）点击 FloppyImageEditor 工具栏上的保存按钮，关闭该工具。



启动调试，待 EOS 启动完毕，在 EOS 控制台中输入命令“dir”后按回车。



4. 实验的思考与问题分析

- 4.1. 在 ConsoleCmdScanDisk 函数中扫描 FAT 表时，为什么不使用 FAT 表项的数量进行计数，而是使用簇的数量进行计数呢？而且为什么簇的数量要从 2 开始计数呢？

答：文件分配表（File Allocation Table）用于将数据区中的磁盘空间分配给文件，属于典型的显式链接方式。文件分配表被划分为紧密排列的若干个表项，每个表项都与数据区中的一个簇相对应，而且表项的序号也是与簇号一一对应的，本来序号为 0 和 1 的 FAT 表项应该对应于簇 0 和簇 1，但是由于这两个表项被设置成了固定值，簇 0 和簇 1 就没有存在的意义了。

- 4.2. 在 ConsoleCmdScanDisk 函数中扫描 FAT 表时，统计了空闲簇的数量，然后使用簇的总数减去空闲簇的数量做为占用簇的数量，这种做法正确吗？是否还有其他类型的簇没有考虑到呢？修改 ConsoleCmdScanDisk 函数，统计出各种类型簇的数量。

答：不正确，还有坏簇。

```

fprintf(StdHandle, "Number Of Clusters: %d\n",
pVcb->Bpb.Sectors - ((pVcb->Bpb.ReservedSectors + pVcb->Bpb.Fats * pVcb->Bp
b.SectorsPerFat)));

```

- 4.3. 在 FAT12 文件系统中，删除一个文件只是将文件对应的目录项中文件名的

第一个字节修改为 0xE5，尝试修改“dir”命令函数 ConsoleCmdDir 的源代码，不但能够输出现有文件的信息，还能够输出已经被删除文件的信息，被删除文件的信息可以包括文件名、大小、最后改写日期、起始簇号等信息。考虑一下这种删除文件方式的优点和缺点。

答：

```
//  
// 跳过未使用的目录项和被删除的目录项  
//  
if(0x0 == pDirEntry->Name[0])  
//  || (CHAR)0xE5 == pDirEntry->Name[0])  
    continue;
```

优点：删除文件迅速，将文件名的第一个字节修改为 0xE5 即可，恢复文件方便，查找文件名的第一个字节为 0xE5 的。

缺点：文件依然占用存储空间。

5. 总结和感想体会

通过本次试验，我明白了 FAT12 文件系统的组成分为引导扇区、文件分配表（FAT）、根目录和数据区，对于文件分配表的指针表项有了一定的了解。学会了 sd 和 dir 命令的使用，以及对相应的函数 ConsoleCmdScanDisk 和 ConsoleCmdDir 进行修改操作。